CHAPITRE I. INTRODUCTION

Objectif : Connaître le but du cours d'algorithmique

Éléments de contenu :

- Qu'est ce qu'une application informatique ?
- Comment arriver d'un problème réel à un programme pouvant être exécuté par ordinateur
- Liens entre ALGORITHMIQUE et STRUCTURES DE DONNEES

I.1.Intérêt de l'algorithmique

Informatiser une application, facturation de la consommation d'eau, par exemple, c'est faire réaliser par ordinateur, une tâche qui était réalisée par l'Homme.
Pour faire exécuter une tâche par ordinateur, il faut tout d'abord, détailler suffisamment les étapes de résolution du problème, pour qu'elle soit exécutable par l'homme. Ensuite, transférer la résolution en ***une suite d'étapes si élémentaire et simple à exécuter***, pouvant être codée en ***un programme*** dans un langage compréhensible par ordinateur.
Toute ***suite d'étapes si élémentaire et simple à exécuter*** s'appelle un ALGORITHME.

Un ***programme*** c'est un algorithme codé dans un langage compréhensible par ordinateur à l'aide d'un compilateur (traducteur).

I.2.Définitions

L'algorithme *est le résultat d'une démarche logique de résolution d'un problème pour la mise en œuvre pratique sur ordinateur et afin d'obtenir des résultats concrets il faut passer par l'intermédiaire d'un langage de propagation.*

Un algorithme *décrit une succession d'opérations qui, si elles sont fidèlement exécutées, produiront le résultat désiré.*

Un algorithme *est une suite d'actions que devra effectuer un automate pour arriver en un temps fini, à un résultat déterminé à partir d'une situation donnée. La suite d'opérations sera composée d'actions élémentaires appelées* ***instructions****.*

1-a) Qu'est ce que l'Algorithmique ?

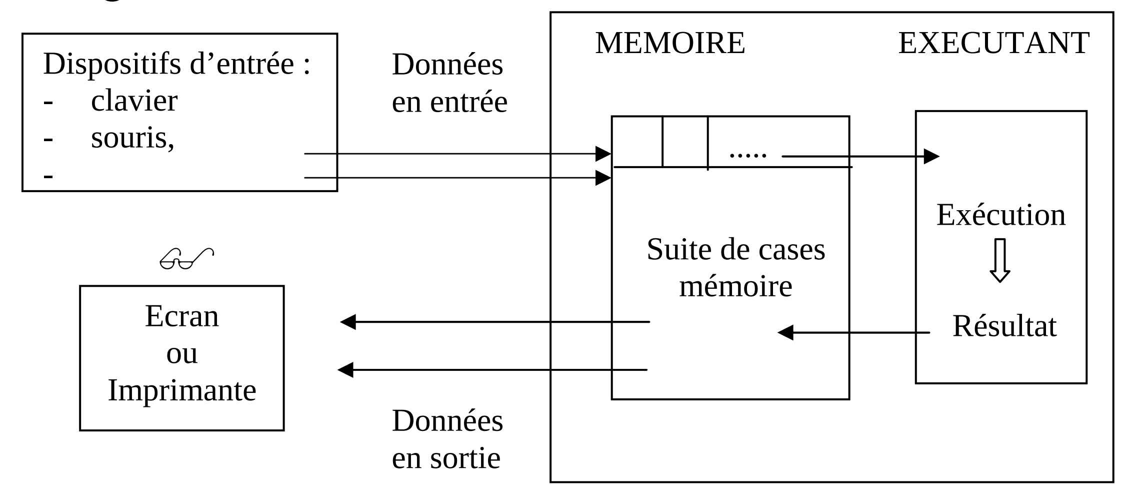
C'est la logique d'écrire des algorithmes. Pour pouvoir écrire des algorithmes, il faut connaître la résolution manuelle du problème, connaître les capacités de l'ordinateur en terme d'actions élémentaires qu'il peut assurer et la logique d'exécution des instructions.

I.3.Les étapes de résolution d'un problème

1. Comprendre l'énoncé du problème
2. Décomposer le problème en sous-problèmes plus simple à résoudre
3. Associer à chaque sous problème, une spécification :
   - Les données nécessaires
   - Les données résultantes
   - La démarche à suivre pour arriver au résultat en partant d'un ensemble de données.
4. Elaboration d'un algorithme.

Illustration du fonctionnement d'un ordinateur





On peut dire que la partie EXECUTANT est le problème de l'algorithmique, et la partie MEMOIRE (stockage de donnée) concerne la matière " Structures de données ".

I.4.Structure d'un algorithme

```
ALGORITHME nom_de_l'algorithme
CONST {Définition des constantes}
TYPE {Définition de types}
VAR   {Déclaration de variables}
DEBUT
     {Suite d'instructions}
FIN
```

1-b) ***Rappel des notions de :***

- Constante,
- Type,
- Variable.

1-c)

1-d) ***Exemple 1***

ALGORITHME afficher
DEBUT
    Ecrire("La valeur de 3*5 est ", 3*5)
FIN

Cet algorithme permet d'afficher sur l'écran la phrase suivante :
La valeur de 3*5 est 15

Exemple 2

On veut écrire l'algorithme qui permet de saisir 3 notes d'un étudiant dans trois matières, étant donnés les coefficients respectifs 2, 3 et 1.

Résolution

A partir de l'énoncé du problème, nous recherchons la solution par une démarche en 2 phases.

- On doit comprendre comment le résoudre manuellement,
- Définir ce qu'on a besoin comme ***données***, quelles est la démarche à suivre (***formules de calcul)*** pour arriver aux ***résultats.***

Pour notre problème, nous connaissons les coefficients et la formule de calcul ($\sum N_i*C_i$ / $\sum C_i$), nous avons besoins des notes de chaque matière ***dans l'ordre,*** et enfin nous pouvons communiquer le résultat à l'utilisateur.

```
ALGORITHME MOYENNE
CONST C1=2
      C2=3
      C3=1
VAR
        N1, N2, N3 : REEL
        MOY : REEL
DEBUT
{Affichage message : Invitation de l'utilisateur à introduire des données}
        ECRIRE(" Donner trois valeurs réelles ")
{Saisie des notes}
        LIRE(N1, N2, N3)
{Calcul de la moyenne}
        MOY← (N1*C1+N2*C2+N3*C3) / (C1+C2+C3)
{Affichage du résultat sur l'écran}
        ECRIRE(" La moyenne est = ", MOY)
FIN
```

Remarque : Le texte entre les accolades est purement explicatif, il sert à rendre l'algorithme plus lisible.

CHAPITRE II. LES ACTIONS ALGORITHMIQUES SIMPLES

Objectif : Comprendre les actions algorithmiques simples et connaître leurs syntaxes

Éléments de contenu :

- Concepts de base
- La saisie de données
- L'affichage
- L'affectation
- L'évaluation d'une expression arithmétique

II.1.0. Concepts de base

Dans tout ce qui suit, pour présenter les syntaxes, on suit les règles suivantes :

- Ce qui est entre les crochets est optionnel.
- La suite des points de suspensions "…" veut dire que ce qui précède peut se répéter plusieurs fois.
- Le symbole " | " veut dire : " ou bien ".
- Les mots en majuscule sont des mots réservés.
- Ce qui est entre accolades est un commentaire, pour la lisibilité des algorithmes.

II.2.1. L'affichage : ECRIRE

Cette action permet de communiquer un résultat ou un message sur écran ou sur imprimante pour l'utilisateur.

<u>Syntaxe</u>

ECRIRE(paramètre1 [[,paramètre2]…])
Paramètre = variable | expression | constante
Constante = nombre | message

1-e) Exemples

```
ECRIRE(" La valeur de 3*2 est égale à ", 3*2)
                         ↑                  ↑
                      message           expression

ECRIRE(" La moyenne est = ", MOY)
                              ↑
                          Variable
```

II.3.2. La saisie des données : LIRE

L'ordre LIRE permet à l'ordinateur d'acquérir des données à partir de l'utilisateur, dans des cases mémoire bien définies (qui sont les variables déclarées).

Rappel

Les variables sont des cases mémoire, supposées contenir un type de données, nommées par le nom de variable.

<u>Syntaxe</u>
LIRE(variable1 [[, variable2] …])

Remarques :
1. La saisie se fait uniquement dans des variables. Ce sont les cases (cellules) qui pourront accueillir les données correspondantes.
2. La donnée à introduire doit être de même type que la variable réceptrice.

II.4.3. Les expressions arithmétiques

Parmi les opérateurs, on distingue les fonctions et les opérateurs.
<u>Les fonctions</u>
- La fonction **DIV** permet de donner le résultat de la division entière d'un nombre par un autre. 7 DIV 2 → 3
- La fonction **MOD** (se lit Modulo), permet de donner le reste de la division entière d'un entier par un autre. 7 MOD 2 → 1
- La fonction ****** ou ^ permet d'élever un nombre à la puissance d'un autre. 2**3 →8

<u>Les opérateurs</u>
- Sont le "+", "-", "/", "*" et le "-" un aire.

<u>Ordre de priorité</u>
Les opérateurs suivants sont ordonnés du plus prioritaire au moins prioritaire dans l'évaluation d'une expression arithmétique.
1- Les parenthèses
2- "- " un aire
3- Les fonctions
4- Les opérateurs de multiplication " * " et de division " / "
5- Les opérateurs d'addition " + " et de soustraction " - "

Remarque
Si l'ordre entre les opérateurs dans une expression est le même, on évalue l'expression de gauche à droite.

Exemples
3**2+4 = 9+4=13
3**(2+4)=3**6 car les parenthèses sont plus prioritaires
17 MOD 10 DIV 3=(17MOD10)DIV3=7DIV3=2

II.5.4. L'affectation

C'est l'action de charger une valeur dans une variable. Cette valeur peut elle-même être une variable, le résultat d'une expression arithmétique ou logique ou une constante.

<u>Syntaxe</u>
Variable1 ← variable2 | expression | constante

A ← B **se lit** " A reçoit B "

Le résultat de cette action est de mettre le contenu de la variable B dans la variable A. Si B était une expression, elle aurait été évaluée, ensuite sa valeur est transférée dans la variable réceptrice (à notre gauche).

Remarque

L'affectation ne vide pas la variable émettrice (à notre droite) de sa valeur. Par contre, le contenu de la variable réceptrice est écrasé et remplacé par la nouvelle valeur.

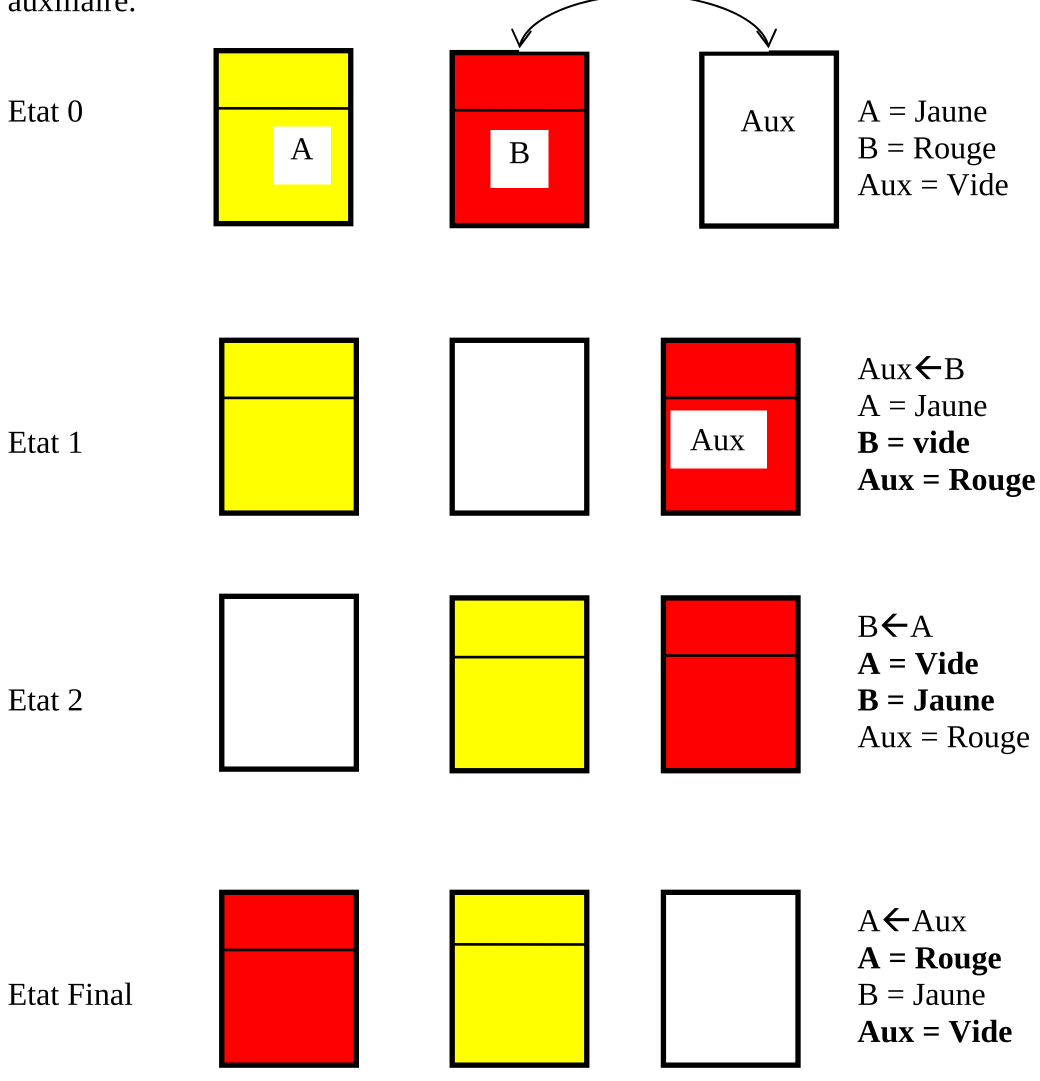
Illustration de l'affectation

Supposons qu'on ait deux récipients A et B où A contient un liquide coloré en jaune et B contient un liquide rouge.

Peut-on échanger les contenus de A et de B (c.-à-d. mettre le liquide rouge dans A et le liquide jaune dans B).

Résultat

Cette opération n'est possible que si on utilise un troisième récipient qu'on appelle récipient auxiliaire.



Avec des variables réelles, cette opération d'échange de contenu se fait entre cases mémoires qui représentent les conteneurs (récipients).

Problème : Echanger les valeurs de 2 variables numériques.

Principe : pour éviter de perdre l'une des 2 valeurs initiales (A et B), on utilise une 3ième variable pour préserver la valeur initiale de la première variable modifiée.

Remarques Importantes

- Toute variable utilisée dans un algorithme doit être déclarée au début de l'algorithme, une fois et une seule.
- L'affectation de valeur à une variable peut être effectuée autant de fois que l'on veut au cours d'un algorithme. La valeur de la variable sera alors modifiée à chaque affectation.
- Lorsqu'une variable apparaît en partie droite d'une action d'affectation, c'est que l'on suppose qu'elle contient obligatoirement une valeur. Cette valeur devra lui avoir été affectée auparavant (par initialisation ou saisie), sinon l'on dira que la valeur est indéfinie, et le résultat de l'affectation ne sera pas défini.
- La variable réceptrice d'une affectation doit être de même type que de la valeur à affecter ou de type compatible. Le type est dit compatible s'il est inclus dans le type de la variable réceptrice. Exemple : REEL ← ENTIER est possible mais pas l'inverse.

Exemple

Ecrire l'algorithme qui permet de calculer le discriminant Δ (delta) d'une équation du second degré.

TD ALGORITHMIQUE I

Les actions simples

Exercice 1

Soit l'algorithme suivant :

ALGORITHME EQUATION2D
VAR a,b,c : REEL
        ~~delta : REEL~~
DEBUT
  Ecrire("Donnez la valeur du premier paramètre")
  Lire(a)
  Ecrire("Donnez la valeur du second paramètre")
  Lire(b)
  Ecrire("Donnez la valeur du troisième paramètre")
  Lire(c)
  delta ← b^2 ~~b * b~~ - 4a * c
  Ecrire(" le discriminant est = Δ ")
~~Fin~~

1 - Décrire cet algorithme en détail (ligne par ligne), en donnant les éventuelles erreurs.
2 - Quelles sont les valeurs de delta dans les cas suivants :
        a=2 b=-3 c=1
        a=1 b=2 c=2

Exercice 2

Ecrire l'algorithme permettant de saisir l'abscisse d'un point A et de calculer son ordonné $f(x)= 2\,x^3 - 3x^2 + 4$
Evaluer le résultat en expliquant les ordres de priorité pour x=-2.

Exercice 3

Ecrire l'algorithme qui permet de permuter les valeurs de A et B sans utiliser de variable auxiliaire.

Exercice 4

Faire l'algorithme qui lit les coordonnées de deux vecteurs u et v, et de calculer leur norme et leur produit scalaire.

Exercice 5

Ecrire l'algorithme qui permet de saisir les paramètres d'une équation du second degré et de calculer son discriminant Δ.

Exercice 6

Ecrire l'algorithme permettant de calculer et d'afficher le salaire net d'un employé. Sachant que :

- Le salaire net = Salaire brut – Valeur de l'impôt – Valeur de CNSS
- Salaire brut = (Salaire de base + Prime de technicité + Prime de transport + Prime des enfants) * Taux de travail
- Taux de travail = Nombre de jours travaillés / 26

- Prime des enfants = Prime d'un enfant * Nombre d'enfants
- Valeur de l'Impôt = Taux de l'Impôt * Salaire Brut
- Valeur de CNSS = Taux de CNSS * Salaire Brut
- Taux CNSS = 26,5%
- Taux Impôt = 2%

Indication :
Décrire l'environnement de travail : toutes les variables en entrée, en sortie et de calcul.

CHAPITRE III. LES STRUCTURES CONDITIONNELLES

III.1. Introduction

Souvent les problèmes nécessitent l'étude de plusieurs situations qui ne peuvent pas être traitées par les séquences d'actions simples. Puisqu'on a plusieurs situations, et qu'avant l'exécution, on ne sait pas à quel cas de figure on aura à exécuter, dans l'algorithme on doit prévoir tous les cas possibles.
Ce sont les ***structures conditionnelles*** qui le permettent, en se basant sur ce qu'on appelle ***prédicat*** ou ***condition***.

III.2. Notion de PREDICAT

Un prédicat est un énoncé ou proposition qui peut être vrai ou faux selon ce qu'on est entrain de parler.
En mathématiques, c'est une expression contenant une ou plusieurs variables et qui est susceptible de devenir une proposition vraie ou fausse selon les valeurs attribuées à ces variables.
Exemple :
(10 < 15) est un prédicat vrai
(10 < 3) est un prédicat faux

III.3. Evaluation d'une expression logique

Une condition est une expression de type logique. Ils lui correspondent deux valeurs possibles VRAI et FAUX qu'on note par V ou F.
Considérons deux variables logiques A et B. Voyons quels sont les opérateurs logiques et leurs ordres de priorités.

1-f) Notons que

- ✓ (A = faux) ⇔ non A
- ✓ (A = vrai) ⇔ A

Les opérateurs logiques sont :

- ✓ La négation : "non"
- ✓ L'intersection : "et"
- ✓ L'union : "ou"

1-g) Notation et Ordre de priorité des opérateurs logiques

1. non : $\neg$
2. et : $\wedge$
3. ou : $\vee$

1-h)

1-i) Tableaux d'évaluations

La négation d'une condition

A	Non A
Vrai	Faux
Faux	Vrai

L'intersection de deux conditions

A et B	Vrai	Faux
Vrai	Vrai	Faux
Faux	Faux	Faux

L'union de deux conditions

A ou B	Vrai	Faux
Vrai	Vrai	Vrai
Faux	Vrai	Faux

Théorème de DE MORGAN

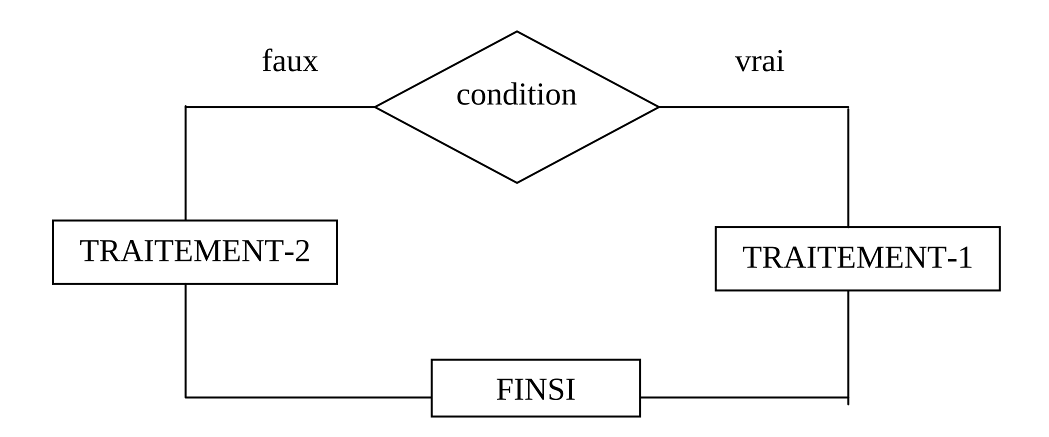
- ✓ $\neg(A \wedge B) \Leftrightarrow \neg A \vee \neg B$
- ✓ $\neg(A \vee B) \Leftrightarrow \neg A \wedge \neg B$

III.4. La structure conditionnelle SI

Syntaxe

SI <Condition> ALORS
    <suite d'action(s)-1>
[SINON
    <suite d'actions(s)-2>]
FINSI

Format Organigramme



- La <condition> est un prédicat, qui peut être vrai ou faux, selon les valeurs des paramètres la constituant.
- Si la condition est vérifiée (sa valeur est vrai), c'est la <suite d'actions-1> qui sera exécutée. Ensuite, le système passe à l'exécution juste après le FINSI.
- Dans le cas contraire, lorsque la condition n'est pas vérifiée (valeur de la condition est faux), c'est la <suite d'actions-2> qui s'exécute, en cas où celle ci existe (facultative). Si elle n'existe pas, le système passe directement à l'instruction qui suit le FINSI.
- Les suites d'actions 1 et 2, peuvent être des actions simples ou même des structures conditionnelles.

1-j) Exemple 1

Lire un nombre réel, et dire s’il est positif ou strictement négatif.

```
ALGORITHME POS-NEG
VAR A : réel
DEBUT
     ECRIRE("Donner un nombre ")
     LIRE(A)
     SI (A < 0) ALORS
          ECRIRE(A, " est négatif ")
     SINON
          ECRIRE(A, " est positif ")
     FINSI
FIN
```

Autrement :

```
ALGORITHME POS-NEG-1
VAR A : réel
     B : logique
DEBUT
     ECRIRE("Donner un nombre ")
     LIRE(A)
     B ← (A < 0)
     SI (B) ALORS
          ECRIRE(A, " est négatif ")
     SINON
          ECRIRE(A, " est positif ")
     FINSI
FIN
```

Dans cet exemple, on a déterminé uniquement les cas de positivité ou de négativité, et on n'a pas déterminé le cas où A est nulle.

```
ALGORITHME POS-NEG-NUL
VAR A : réel
DEBUT
     ECRIRE("Donner un nombre ")
     LIRE(A)
     SI (A < 0) ALORS
          ECRIRE(A, " est négatif ")
     SINON {A >= 0}
       SI (A > 0)ALORS
          ECRIRE(A, " est positif ")
       SINON {A = 0}
          ECRIRE (A, "est nulle")
       FINSI
     FINSI
FIN
```

1-k) Exemples

1) Ecrire l'algorithme qui permet de déterminer si un entier lu est pair ou impair.
2) Ecrire l'algorithme qui permet de saisir deux nombres A et B et de déterminer si la valeur de A est supérieure, inférieure ou égale à B.

III.5. La structure conditionnelle SELON

Cette structure conditionnelle est appelée aussi ***à choix multiple*** ou ***sélective*** car elle sélectionne entre plusieurs choix à la fois, et non entre deux choix alternatifs (le cas de la structure SI).

Syntaxe

SELON (sélecteur) FAIRE
Cas <liste de valeurs-1> : <suite d'action (s)-1>
[Cas <liste de valeur-2> : <suite d'action (s)-2>
……….]
[SINON : <suite d'action (s)-n>]
FINSELON

Le ***sélecteur*** peut être une variable de type scalaire ou une expression arithmétique ou logique.

La structure SELON évalue le "sélecteur", passe à comparer celui ci respectivement avec les valeurs dans les listes. En cas d'égalité avec une valeur, les actions correspondantes, qui sont devant cette valeur seront exécutées.

Devant "Cas", il peut y avoir une seule valeur, une suite de valeurs séparées par des virgules et/ou un intervalle de valeurs.

Après avoir traiter la suite d'actions correspondante, l'exécution se poursuit après le FINSELON.

Remarque

1. Le sélecteur doit avoir le même type que les valeurs devant les cas.
2. Le type de ces valeurs ne doit être, ni réel ni chaîne de caractères.

1-l) Exemple

Ecrire l'algorithme qui permet de saisir un numéro de couleur de l'arc-en-ciel et d'afficher la couleur correspondante : 1: rouge, 2 : orangé, 3 : jaune, 4 : vert, 5 : bleu, 6 : indigo et 7 : violet.

TD ALGORITHMIQUE I

Les Structures Conditionnelles

Exercice 1

Evaluer les expressions logiques suivantes, avec (a, b, c, d) = (2, 3,5, 10) et (X, Y) = (V, F).

1) $(a < b) \wedge (a < c)$	2) $\neg ((a < b) \wedge (a < c))$	3) $\neg (a < b) \wedge (a < c)$
4) $(a < c) \wedge (c = d/2)$	5) $(d / a = c) = Y$	6) $(d / c = b) = Y$
7) $(d / c = b) = X$	8) $(a < b) \wedge (d < c)$	9) $(a < b) \wedge (d < c) = X$

Exercice 2

Réécrire l'exercice 6 de la série N°1 en supposant que le taux de l'impôt n'est pas fixe mais il varie selon la valeur du salaire de base. En effet :

- ✓ Taux de l'impôt = 0 si le salaire de base < 150
- ✓ Taux de l'impôt = 2% si le salaire de base ∈[150,250[
- ✓ Taux de l'impôt = 5% si le salaire de base ∈[250,500[
- ✓ Taux de l'impôt = 12% si le salaire de base >= 500.

En plus, la prime des enfants est définit comme suit :

- ✓ 7DT pour le premier enfant,
- ✓ 5DT pour le deuxième enfant,
- ✓ 4DT pour le troisième enfant.
- ✓ Pas de prime pour le reste.

Exercice 3

Ecrire l'algorithme qui permet de saisir un nombre puis déterminer s'il appartient à un intervalle donné, sachant que les extrémités de l'intervalle sont fixées par l'utilisateur.

Exercice 4

Ecrire l'algorithme qui permet de calculer le montant des heures supplémentaires d'un employé, sachant le prix unitaire d'une heure selon le barème suivant :

- Les 39 premières heures sans supplément,
- De la 40ième à la 44ième heure sont majorées de 50%,
- De la 45ième à la 49ième heure sont majorées de 75%,
- De la 50ième heure ou plus, sont majorées de 100%.

Exercice 5

Ecrire l'algorithme qui permet de saisir la moyenne générale d'un étudiant et de déterminer son résultat et sa mention. (les conditions de rachat sont appliquées à partir de 9,75.

Exercice 6

Ecrire l'algorithme qui permet de saisir les trois paramètres d'une équation du second degré, et de discuter les solutions selon les valeurs de a, b et c, lorsqu'elles sont nulles ou pas.

Exercice 7

Ecrire l'algorithme qui permet de saisir le jour, le mois et l'année d'une date (Mois : numéro du mois), et de déterminer si elle est correcte ou non, et où est l'erreur.

Exercice 8

Ecrire l'algorithme qui permet de saisir deux nombres, et un opérateur et d'évaluer l'expression arithmétique correspondante.

Exercice 9

Ecrire l'algorithme CONTRAT qui permet d'aider une compagnie d'assurance à prendre une décision concernant les demandes d'affiliation en se basant sur les critères suivants :

CRITERE DECISION	AGE	Bonne santé	Accident
Contrat A	<=30	OUI	NON
Contrat B	>30	OUI	OUI
Contrat refusé	-	NON	OUI
Expertise demandée	-	OUI	OUI

Exercice 10

Ecrire un algorithme qui permet de saisir un numéro de mois et un jour (le contrôle n'est pas demandé) et d'afficher la période correspondante selon le tableau suivant :

Période	**DU**	**AU**
Vacances d'été	1/7	15/9
Premier trimestre	16/9	19/12
Vacances d'hiver	20/12	3/1
Deuxième trimestre	4/1	19/3
Vacances de printemps	20/3	3 / 4
Troisième trimestre	4/4	30/6

Exercice 11

Ecrire l'algorithme permettant de lire la valeur de la variable DEVINETTE et d'afficher parmi les messages suivants celui qui correspond à la valeur trouvée :
ROUGE si la couleur vaut R ou r
VERT si la couleur vaut V ou v
BLEU si la couleur vaut B ou b
NOIR pour tout autre caractère.

Exercice 12

Ecrire l'algorithme permettant de lire la valeur de la température de l'eau et d'afficher son état :
GLACE Si la température inférieure à 0,
EAU Si la température est strictement supérieure à 0 et inférieure à 100,
VAPEUR Si la température supérieure à 100.

Exercice 13

Ecrire l'algorithme qui lit un entier positif inférieur à 999 (composé de trois chiffres au maximum) et d'afficher le nombre de centaines, de dizaines et d'unités.

CHAPITRE IV. LES STRUCTURES REPETITIVES

IV.1. Introduction

Dans les problèmes quotidiens, on ne traite pas uniquement des séquences d'actions, sous ou sans conditions, mais il peut être fréquent d'être obligé d'exécuter un traitement (séquence d'actions), plusieurs fois. En effet, pour saisir les N notes d'un étudiant et calculer sa moyenne, on est amené à saisir N variables, puis faire la somme et ensuite diviser la somme par N. Cette solution nécessite la réservation de l'espace par la déclaration des variables, et une série de séquences d'écriture/lecture. Ce problème est résolu à l'aide des structures répétitives. Celles ci permettent de donner un ordre de répétition d'une action ou d'une séquence d'actions une ou plusieurs fois.

IV.2. La boucle POUR

Cette structure exprime la répétition d'un traitement un nombre de fois.

1-m)

1-n) Syntaxe

POUR Vc DE Vi A Vf [PAS Vp] FAIRE
<Traitement>
FINFAIRE

Où Vc est une variable entière, qui compte le nombre de répétition du <Traitement>,
Vi la valeur initiale à laquelle Vc est initialisé,
Vf la valeur finale à laquelle se termine Vc,
Vp la valeur du pas, c'est la valeur qu'on rajoute à Vc à chaque fin de traitement.

Remarque

1. La boucle POUR est utilisée lorsqu'on connaît le nombre de répétition du <Traitement> d'avance.
2. La valeur du pas peut être positive ou négative et par conséquent, il faut; au départ de la boucle; que Vi <= Vf ou Vi >= Vf selon la positivité ou la négativité de cette valeur.
3. La valeur du pas est égale à 1 par défaut.

Les étapes d'exécution de la boucle POUR

1) Initialisation de Vc par la valeur de Vi (comme si on avait Vc←Vi)
2) Test si Vi dépasse (±) Vf (du côté supérieur ou inférieur, selon la positivité ou la négativité du pas).
   Si oui, alors la boucle s'arrête et l'exécution se poursuit après le FINFAIRE
   Sinon,
   - Exécution du <Traitement>,
   - Incrémentation ou décrémentation de Vc par la valeur du pas,
   - Retour à l'étape 2.

Application

Ecrire l'algorithme qui permet de saisir les moyennes des N étudiants de la classe Informatique et de calculer la moyenne générale de la classe.

Résolution

Sans les boucles, on est obligé de déclarer N variables, et d'écrire N actions LIRE.

N fois

LIRE(note)
S← S + MOY
LIRE(MOY)
S← S + MOY

…..
LIRE(MOY)
S← S + MOY

La boucle POUR donne l'ordre à la machine d'itérer les deux actions LIRE(MOY) S←S+ MOY N fois.
Donc le compteur varie de 1 jusqu'à N avec un pas de 1.

```
ALGORITHME MOYENNE
VAR i, N : entier
     MOY, MC : réel
DEBUT
ECRIRE("Donner le nombre d'étudiants")
LIRE(N)
SI (N > 0) ALORS
     S ← 0 {Initialisation de S}
     POUR i DE 1 A N FAIRE {Le pas égale 1 par défaut}
          ECRIRE("Donner la moyenne de l'étudiant n°", i)
          LIRE(MOY)
          S ← S + MOY {on rajoute la moyenne du ième étudiant à la somme}
     FIN FAIRE
     MC ← S / N
     ECRIRE("La moyenne de la classe est : ", MC)
SINON
     ECRIRE("Erreur dans le nombre d'étudiants")
FINSI
FIN
```

Remarque Juste Avant le FIN FAIRE, le changement de la valeur de i se fait automatiquement.

Application 1

Ecrire l'algorithme qui permet d'afficher tous les nombres pairs qui existent entre 1 et 10.

1ière solution

POUR i de 2 à 10 pas 2 Faire
    ECRIRE(i)
FINFAIRE

2ième solution

POUR i de 2 à 10 Faire
SI (i mod 2 = 0) ALORS
    ECRIRE(i)
FINSI
FINFAIRE

3ième solution

POUR i de 1 à 5 Faire
    ECRIRE(2*i)
FINFAIRE

Application 2

Ecrire l'algorithme qui permet d'afficher tous les nombres impairs entre 50 et 100 dans l'ordre décroissant.

```
POUR i de 99 à 50 PAS (-2) FAIRE
        ECRIRE(i)
FIN FAIRE
```

La valeur finale peut être 50 ou 51 car le test de sortie est i < Vf (49 < 50 ou à 51)

IV.3. La boucle Répéter … Jusqu'à

Syntaxe

Répéter
        <Traitement>
Jusqu'à (condition d'arrêt)

Cet ordre d'itération permet de répéter le <Traitement> une ou plusieurs fois et de s'arrêter sur une condition. En effet, lorsque la condition est vérifiée, la boucle s'arrête, si non elle ré-exécute le <Traitement>.

Remarques

1. Dans cette boucle, le traitement est exécuté au moins une fois avant l'évaluation de la condition d'arrêt.
2. Il doit y avoir une action dans le <Traitement> qui modifie la valeur de la condition.

Les étapes d'exécution de la boucle Répéter

1) Exécution du <Traitement>
2) Test de la valeur de la <condition d'arrêt>
   Si elle est vérifiée Alors la boucle s'arrête
   Sinon Retour à l'étape 1.

Application

Ecrire un algorithme qui saisit un nombre pair et qui détermine combien de fois il est divisible par 2. Exemple 8 est divisible 3 fois par 2 (2*2*2).

```
ALGORITHME PAIR-NBDIV2
VAR N, N2 : entier
DEBUT
{Saisie d'un entier qui doit être pair}
Répéter
     ECRIRE("Donner un entier pair")
     LIRE(N)
Jusqu'à (N MOD 2 = 0) {condition pour que N soit pair}
{Détermination du nombre de division par 2}
N2 ← 0
NB ← N
Répéter
     NB ← NB div 2
     N2 ← N2 +1
Jusqu'à (NB MOD 2 <> 0) {On s'arrête lorsque NB n'est plus divisible par 2}

ECRIRE(N, "est divisible par 2", N2,"fois")
FIN
```

IV.4. La boucle TANT QUE …

<u>Syntaxe</u>

TANT QUE (condition d'exécution)
FAIRE
    <Traitement>
FIN FAIRE

Cet ordre d'itération permet de répéter le <Traitement> **zéro** ou plusieurs fois et de s'arrêter lorsque la condition d'exécution n'est plus vérifiée. En effet, lorsque la condition d'exécution est vérifiée, le <Traitement> est exécuté, si non elle s'arrête.

<u>Les étapes d'exécution de la boucle Répéter</u>

1) Test de la valeur de la <condition d'exécution>
2) Si elle est vérifiée Alors
    Exécution du <Traitement>
    Retour à l'étape 1.
   Sinon Arrêt de la boucle.

Remarques

1. Dans cette boucle, le traitement peut ne pas être exécuté aucune fois, c'est lorsque la condition d'exécution est à faux dés le départ.
2. Les paramètres de la condition doivent être initialisés par lecture ou par affectation avant la boucle.
3. Il doit y avoir une action dans le <Traitement> qui modifie la valeur de la condition.

<u>Application</u>

Ecrire un algorithme qui saisit un nombre pair et qui détermine combien de fois il est divisible par 2. Exemple 8 est divisible 3 fois par 2 (2*2*2).

```
ALGORITHME PAIR-NBDIV2
VAR N, N2 : entier
DEBUT
{Saisie d'un entier qui doit être pair}
Répéter
     ECRIRE("Donner un entier pair")
     LIRE(N)
Jusqu'à (N MOD 2 = 0) {condition pour que N soit pair}
{Détermination du nombre de division par 2}
N2 ← 0
NB ← N
TANT QUE (NB MOD 2 = 0)
FAIRE
     NB ← NB div 2
     N2 ← N2 +1
FIN FAIRE {On s'arrête lorsque NB n'est plus divisible par 2}
ECRIRE(N, "est divisible par 2", N2,"fois")
FIN
```

- La condition d'arrêt avec la boucle Répéter est l'inverse de la condition d'exécution de la boucle TANTQUE.

Remarque
Le Traitement d'une boucle peut contenir lui aussi une autre boucle. On l'appelle dans ce cas des boucles imbriquées.

TD ALGORITHMIQUE I

Les structures répétitives

Exercice 1

Ecrire l'algorithme qui permet d'afficher les N premiers entiers impairs dans l'ordre décroissant.

Exercice 2

Ecrire l'algorithme qui permet d'afficher les diviseurs d'un entiers N.

Exercice 3

Ecrire l'algorithme qui détermine si une entier N est parfait ou non. Un entier est dit parfait s'il est égal à la somme de ses diviseurs. Exemple 6 = 3 + 2 +1

Exercice 4

Ecrire l'algorithme qui permet de calculer le produit de deux entiers en utilisant des additions successives.

Exercice 5

Ecrire l'algorithme qui permet de calculer la division de deux entiers en utilisant des soustractions successives

Exercice 6

Ecrire l'algorithme qui permet de saisir un entier N et d'afficher s'il est premier ou non. Un nombre est dit premier s'il est divisible uniquement par 1 et par lui-même.

Exercice 7

Ecrire l'algorithme qui détermine le 20ième terme d'une suite définie par :

$S_0 = 2$, $S_1 = 3$ et $S_n = S_{n-2} + (-1)^n * S_{n-1}$

Exercice 8

Ecrire l'algorithme qui détermine le $N^{ième}$ terme d'une suite définie par :

$S_0 = 2$, $S_1 = 3$, $S_2 = -2$ et $S_n = S_{n-3} + (-1)^n * S_{n-1}$

Exercice 9

On démontre en mathématique que le cosinus d'un angle exprimé en radian est donné par la somme infinie suivante :

$COS(x) = 1 – X^2 / 2! + X^4/4! – X^6/6! +…$

On décide d'arrêter la somme à un certain rang n (n>3) donné.

Ecrire l'algorithme qui permet d'évaluer le cosinus d'une valeur x donnée.

Exercice 10

Ecrire l'algorithme qui permet de saisir autant de nombres que l'utilisateur le veuille, et de déterminer le nombre de réels strictement positifs et celui des négatifs. On s'arrête lorsque la valeur est 999.

Exercice 11

Ecrire l'algorithme qui permet de saisir autant de nombres que l'utilisateur le veuille, pourvu qu'ils soient dans l'ordre croissant. On s'arrête lorsque la valeur est 999.

Exercice 12
Ecrire l'algorithme qui permet de saisir un entier positif en décimal et de le transformer en binaire.
Exemple $(7)_{10} = (111)_2$

Exercice 13
Ecrire un algorithme qui permet de saisir un entier et une base inférieure ou égale à 10 et de vérifier si ce nombre appartient à la base ou non.

Exercice 14
Ecrire un algorithme qui permet de saisir deux entiers et de vérifier si les chiffres du premier appartiennent à ceux du second nombre ou non.

Exercice 15
Ecrire un algorithme qui permet de saisir deux entiers positifs et de déterminer leur plus grand commun diviseur (PGCD).
Le PGCD(A,B) = PGCD(A-B, B) si A est le plus grand et
à PGCD(A,B) = PGCD(A, B-A) si B est le plus grand. Si A=B le PGCD(A,B) est A ou B.

Exercice 16
Ecrire un algorithme qui permet de calculer la factorielle d'un entier N donné.

Exercice 17
Ecrire un algorithme qui permet de saisir des entiers alternatifs (si l'un est positif sont suivant doit être négatif et vice versa).

Exercice 18
Ecrire l'algorithme qui permet de saisir deux entiers et de déterminer leur plus petit commun multiple (PPCM).

CHAPITRE V. TRAITEMENT DES TABLEAUX

Rappel

Pourquoi les tableaux ?

1) Calculer la moyenne de 30 élèves
2) Effectuer leur classement

➷ ***Réponse***

pour i de 1 à 30
faire
    Ecrire (" Donner la moyenne de l'étudiant N°",i)
    Lire (moyenne)
Fin faire

➷ ***Conclusion*** : On ne peut pas effectuer le classement

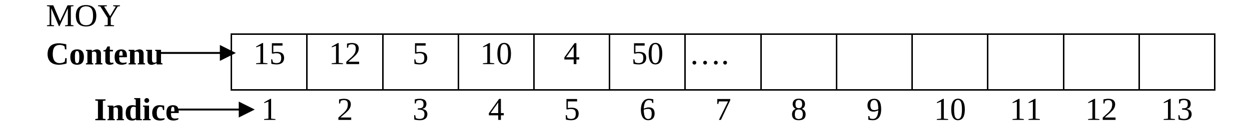
Pourquoi ? Parce qu'on ne garde pas les moyennes précédentes et la variable moyenne contient uniquement la dernière valeur.

Utilisation des tableaux

Intérêt Gain de temps, rétrécissement du volume de l'algorithme et possibilité de réutilisation de toutes les valeurs ultérieurement dans l'algorithme.

Il est plus convenable, alors, de définir un espace mémoire qu'on appelle MOY qui sera divisé en 30 parties équitables, indicées de 1 à 30.

MOY



On définit un tableau de 30 cases à une seule dimension qu'on appelle <u>VECTEUR</u>.

```
ALGORITHME MOYENNE
CONST Bi=1
      Bs=30
VAR T : Tableau [bi..bs] de réel
     i : entier
```

V.1.1. Les vecteurs

Un vecteur est une partie de mémoire contenant n zones variables référencées par le même nom de variable pour accéder à un élément particulier de ce vecteur.

On indice le nom de variable. L'indice peut être une constante, une variable ou une expression arithmétique.

MOY[i]
 ↑ ↖ indice d'un élément du vecteur
 variable qui indique le nom du vecteur

MOY[i] : représente l'élément du vecteur MOY occupant le rang " i ".

L'indice peut être :
- Une constante →MOY[5]
- Une variable →MOY[i]
- Une expression →MOY[i*2]

ATTENTION
Avant d'utiliser un tableau, il faut déclarer sa taille pour que le système réserve la place en mémoire, nécessaire pour stocker tous les éléments de ce tableau.
Les éléments d'un même tableau doivent être de même type.

V.2.Rappel de Déclaration d'un vecteur

Dans la partie CONST, on peut définir la taille du tableau. Ensuite, on peut déclarer le nombre d'éléments à saisir dans le tableau.
Remarque : Le nombre d'éléments à saisir ne doit pas dépasser la taille du tableau pour ne pas déborder sa capacité.
On appelle dimension d'un vecteur le nombre d'éléments qui constituent ce vecteur.

V.3.Chargement d'un Vecteur

Le chargement d'un vecteur consiste à saisir les données des éléments du vecteur. (remplir des cases successives du tableau). On doit utiliser une boucle qui permet de saisir à chaque entrée dans la boucle la $i^{ième}$ case.

```
ALGORITHME          Vecteur
CONST        N = 30
VAR
        MOY : Tableau[1..N] de réels
Début
{ chargement du tableau }
Pour i de 1 à N
Faire
        Ecrire (" donner la moyenne de l'étudiant N° " , i )
        Lire ( MOY [i])
Fin Faire
{ fin chargement }

{Calcul de la somme des moyennes}
SMOY ← 0
Pour i de 1 à N
Faire
        SMOY←SMOY+MOY[i]
Fin Faire

SMOY ← SMOY / 30
Ecrire (" la moyenne du groupe est ", SMOY )
{ calcul de la différence entre la moyenne de groupe et celle  de l'étudiant }
Pour i de 1 à N
Faire
```

Ecrire (" la différence de la moyenne du groupe et celle de l'étudiant ",i , " est= ", SMOY-MOY[i])
Fin Faire
Fin

✍ On peut écrire les deux premières boucle en une seule. Simplifier alors cet algorithme.

Remarque
La taille d'un tableau est fixe et ne peut être donc changée dans un programme : il en résulte deux défauts :
- Si on limite trop la taille d'un tableau on risque le dépassement de capacité.
- La place mémoire réservée est insuffisante pour recevoir toutes les données.

Application
1) Charger un vecteur de 10 éléments par les 10 premiers entiers naturels positifs.
2) Charger un vecteur de 10 éléments par les 10 premiers multiples de 7.

1-o) Recherche dans un vecteur

Recherche séquentielle
On peut chercher le nombre d'apparition d'un élément dans un vecteur, sa ou bien ses positions. Pour cela, on doit parcourir tout le vecteur élément par élément et le comparer avec la valeur de l'élément à chercher.

Applications
1. Chercher la position de la première occurrence d'un élément e dans un vecteur V contenant N éléments. (On suppose que le vecteur est définit)
2. Chercher le nombre d'apparition d'un élément e dans un vecteur V contenant N éléments, ainsi que les positions des occurrences de cet élément.

Réponse 1
```
i ← 1
Trouv ← vrai
Tant que ((i <= N) et (Trouv = vrai))
     Faire
          Si V[i] = e Alors
               Trouv ← Faux
          Sinon
               i ← i +1
          Fin Si
     Fin Faire
Si (Trouv = vrai) Alors
     Ecrire(e, "se trouve à la position" , i)
Sinon
     Ecrire(e, "ne se trouve pas dans V")
Fin Si
```

Recherche dichotomique
Ce type de recherche s'effectue dans un tableau ordonné.
Principe
1. On divise le tableau en deux parties sensiblement égales,
2. On compare la valeur à chercher avec l'élément du milieu,

3. Si elles ne sont pas égales, on s'intéresse uniquement la partie contenant les éléments voulus et on délaisse l'autre partie.
4. On recommence ces 3 étapes jusqu'à avoir un seul élément à comparer.

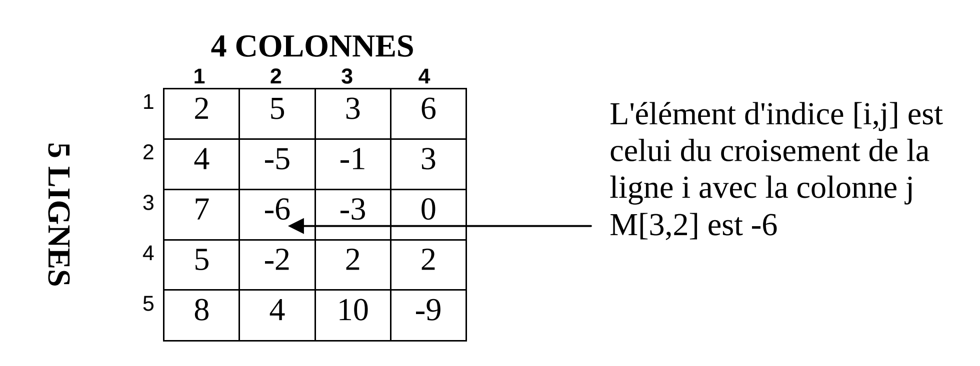
Application
On suppose qu'on dispose d'un vecteur V de N éléments. On veut chercher la valeur Val.

```
ALGORITHME DICHOTHOMIE
...
Inf ← 1
Sup ← N
Trouv  ← vrai
Tant que ((Inf <= Sup) et (Trouv = vrai))
Faire
     Mil← (Inf+Sup)DIV 2
     Si (V[Mil] = Val) Alors
          Trouv← faux
     Sinon
          Si (V[Mil] < Val) Alors
               Inf ← Mil + 1
          Sinon
               Sup ← Mil -1
          Fin Si
     Fin Si
Fin Faire

Si (Trouv = faux) Alors
     Ecrire(Val, "existe à la position" , Mil)
Sinon
     Ecrire(Val, "n'existe pas dans V)
Fin Si
```

V.4.2. Les matrices

Les matrices sont les tableaux à deux dimensions.



	1	2	3	4
1	2	5	3	6
2	4	-5	-1	3
3	7	-6	-3	0
4	5	-2	2	2
5	8	4	10	-9

CHAPITRE VI. TD ALGORITHMIQUE 1

Chercher le plus petit élément dans un vecteur.

1-p) Exercice 12

Saisissez un vecteur de telle façon qu'il soit ordonné.

Soit un tableau NOM dont les éléments sont de type chaîne de caractères. Ce tableau contient les noms des étudiants ordonnés selon le numéro de registre. Et soit le tableau MOY contenant respectivement la moyenne de chaque étudiant selon le même ordre.
Ecrire l'algorithme qui permet de saisir les deux tableaux puis d'afficher le nom de l'étudiant ayant la meilleure moyenne.

CHAPITRE VII. LES ALGORITHMES DE TRI

Dans ce chapitre on présente quelques algorithmes utiles, qui permettent d'ordonner les éléments d'un tableau dans un ordre croissant ou décroissant. L'ordre est par défaut croissant.

Un vecteur est dit trié si V[i] <= V[i+1], $\forall$ i $\in$ [1..n-1]

1. *Tri par sélection*

1-q) Principe

Utiliser un vecteur VT (vecteur trié) comme vecteur résultat. Celui ci contiendra les éléments du vecteur initial dans l'ordre croissant.
Le principe est de :

0- Chercher le plus grand élément dans le vecteur initial V
1- Sélectionner le plus petit élément dans V
2- Le mettre dans son ordre dans le vecteur VT
3- Le remplacer par le plus grand élément dans le vecteur initial (pour qu'il ne sera plus le minimum)
4- Si le nombre d'éléments dans le vecteur résultat n'est pas identique à celui dans le vecteur initial Retourner à l'étape 1
   Sinon on s'arrête.

1-r) Exemple

Soit le vecteur V contenant 4 éléments.

	V				VT			
Au départ	10	15	-1	7				
Phase 1	10	15	15	7	-1			
Phase 2	10	15	15	15	-1	7		
Phase 3	15	15	15	15	-1	7	10	
Phase 4	15	15	15	15	-1	7	10	15

Schéma de l'algorithme

```
ALGORITHME TRI_SELECTION
CONST Bi = 1
      Bs = 10
VAR  V, VT : Tableau[Bi..Bs] de réel
     N, i, j, indmin : entier
     MIN, MAX : réel
DEBUT
{Chargement du vecteur V}
…
{Recherche du maximum}
MAX←V[1]
Pour i de 2 à N
FAIRE
 Si MAX < V[i] Alors
     MAX←V[i]
 FINSI
FINFAIRE
```

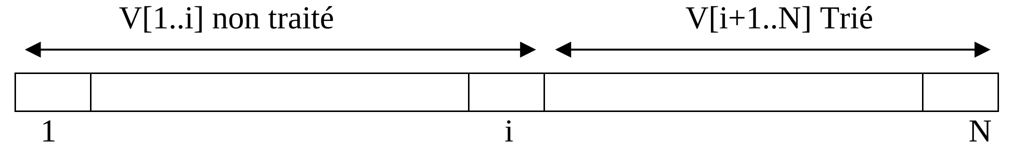
```
POUR i de 1 à N-1 FAIRE
  {Recherche du minimum}
     MIN ← V[1]
     indmin ← 1
     Pour j de 2 à N faire
          Si  MIN > V[j] ALORS
              MIN←V[j]
              Indmin ← j
          Fin si
     Fin Faire
  {Mettre le minimum trouvé à sa place dans le vecteur résultat}
     VT[i] ← MIN
     V[indmin] ← MAX
Fin Faire
VT[N] ← MAX
FIN
```

Peut-on améliorer cet algorithme ?

2. Algorithme de tri par sélection et permutation

Il s'agit ici d'éviter la construction d'un second vecteur et d'utiliser un seul vecteur initial qui sera trié.

Supposons traités n-i (1 <= i < N) éléments du vecteur.



On peut considérer le vecteur V comme la concaténation de deux sous-vecteurs : le sous-vecteur V[1..i] dont les éléments n'ont pas encore été triés, et le sous vecteur V[i+1..N] dont les éléments sont triés. D'autre part tous les éléments du sous-vecteur V[1..i] sont inférieurs ou égaux à l'élément V[i+1].

On a donc :

V[1..i] non traité, V[1..i] <= V[i+1], V[i+1..N] Trié

On a deux cas :

- I = 1
  (V[1] non traité, V[1]<= V[2], V[2..N] trié) donc V[1..N] trié
  L'algorithme est terminé.
- I > 1
  Pour augmenter le sous-vecteur V[i+1..n] d'un élément, il suffit de chercher le plus grand élément contenu dans le sous-vecteur V[1..i] et de placer cet élément en position i.

Schéma de l'algorithme

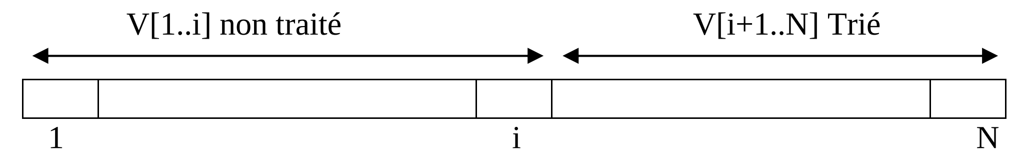
```
ALGORITHME SLECTION_PERMUTATION
CONST Bi = 1
      Bs = 10
```

```
VAR  V : Tableau[Bi..Bs] d'entier
     N, i, j : entier
DEBUT
{Chargement du vecteur V}
…
Pour i de N à 2 Faire
{Recherche de l'indice du maximum dans V[1..i]}
 indmax ← 1
 Pour j de 2 à i
 FAIRE
  Si V[indmax] < V[j] Alors
     indmax ← i
  FIN SI
FIN FAIRE
{Mettre le maximum relatif trouvé à sa place}
Si indmax <> i Alors
     Aux ← V[indmax]
     V[indmax] ← V[i]
     V[i] ← Aux
Fin Si
Fin Faire
```

VII.1.3. Tri par la méthode des bulles

Même principe que le précédent.

Après avoir traité n-i (1 <= i < N) éléments du vecteur.



On peut donc considérer le vecteur V comme la concaténation de deux sous-vecteurs : le sous-vecteur V[1..i] dont les éléments n'ont pas encore été triés, et le sous vecteur V[i+1..N] dont les éléments sont triés. D'autre part tous les éléments du sous-vecteur V[1..i] sont inférieurs ou égaux à l'élément V[i+1].

On a donc :

V[1..i] non traité, V[1..i] <= V[i+1], V[i+1..N] Trié

On a deux cas :

- I = 1
  (V[1] non traité, V[1]<= V[2], V[2..N] trié) donc V[1..N] trié
  L'algorithme est terminé.
- I > 1

Pour augmenter le sous-vecteur V[i+1..n] d'un élément, il suffit de chercher le plus grand élément contenu dans le sous-vecteur V[1..i] et de placer cet élément en position i.

On parcourt le sous-vecteur V[1..i] de gauche à droite et, chaque fois qu'il y a deux éléments consécutifs qui ne sont pas dans l'ordre, on les permute. Cette opération permet d'obtenir en fin du $i^{ième}$ parcours le plus grand élément placé en position i, et les éléments après cette position sont ordonnés.

Schéma de l'algorithme

```
ALGORITHME TRI_BULLE1
CONST N= 10
VAR  V : tableau[1..N] de réel
     N, i, j : entier
     AUX : réel
DEBUT
{Chargement du vecteur}
…
POUR i de N à 2 pas –1 FAIRE
     POUR j de 1 à i FAIRE
          SI V[j]>V[j+1] ALORS
            AUX ← V[j]
            V[j] ← V[j+1]
            V[j+1] ← AUX
          FIN SI
     FIN FAIRE
FIN FAIRE
FIN
```

Application

Exécuter à la main cet algorithme avec les vecteurs suivants :

2	2	-1	3	0	1

1	-1	2	5	13	15

Que remarquez-vous ?

3. Schéma de l'algorithme à bulle optimisé

```
ALGORITHME TRI_BULLE1
CONST N= 10
VAR  V : tableau[1..N] de réel
     N, i, j : entier
     AUX : réel
DEBUT
{Chargement du vecteur}
…
i ← N
atonpermuté ← vrai
TANT QUE (atonpermuté) FAIRE
     j←1
     atonpermuté ← faux
     TANT QUE (j < i) FAIRE
          DEBUT
           SI (V[J+1] < V[j]) ALORS
```

```
               AUX←V[J+1]
               V[J+1] ←V[J]
               V[J] ← AUX
             FIN SI
               atonpermuté←vrai
          FIN
          j←j+1
     FIN
     i←i-1
     FIN
FIN
```

```
POUR i de N à 2 pas –1 FAIRE
     POUR j de 1 à i FAIRE
          SI V[j]>V[j+1] ALORS
            AUX ← V[j]
            V[j] ← V[j+1]
            V[j+1] ← AUX
          FIN SI
     FIN FAIRE
FIN FAIRE
FIN
```

Composer un algorithme lisant une quantité commandée inférieure à 10000 puis calculer le nombre de caisse non nul de chaque type utilisé

VARIABLE LOCALE

Une variable de type locale si elle est définie à l'intérieur de la procédure . Elle n'est accessible que dans l'environnement de la procédure ou elle a été définie.

Lintéret des variables locales c'est qu'elles conntribuent particulierement à une plus grande lisibilité d'un programme (algorithme). Elle minimisent les erreurs.

VARIABLE GLOBAL

Une variable est dite global si elle est définie au niveau de l'algorithme qui appelle la procédure c'est à dire une variable utilisée par la procédure et n'est pas déclarés a l'interieur de cette procédure.

Un mê

me nom de variable peut être glopal ou local

Dans de telle situation le définition d'une variable local détient la présance à l' interieur de son champ d'application

IMPORTANT

Une variable globale peut étre utilisé n'importe où

à l'interieur ou à l'extérieur de la procédure

```
PROCEDURE     MAXI
 DEBUT
  si(A>B)
   alors
    MAX=a
   si non
    MAX=B
 fin si
écrire("le maximum est ,"MAX)
FIN
```

```
ALGORITHME     Maximum
DEBUT
 écrire("donner deux valeurs distinctes")
  REPETER
   lire(A,B)
  JUSQU'A(A<>B)
 MAXI
FIN
```

AetB sont deux variables globales

MAX est une variable local

L'echange d'information entre la procedure et l'algorithme est fait via (atravers) les variables globales

Cette méthode d'echange peut changer le contenu de la variable à l'interieur de la procedure qui peut effecter certaines informations erronées à l'exterieur de la procedure et vice vers ça

Pour resoudre ce probléme on fait recour à l'emploie des parametres qui offrent une meilleur approche à l'echange d'information entre une procedure et son point de reference chaque donnée est transferée entre paramétre réel et un paramétre formel

PARAMETRE FORMEL:
Est un paramétre (variable) défini à l' intérieur de la procedure
exemple:procédure saisie(A,B) ;A,B:paramétres formels
PARAMETRE REEL:
Est un paramétre inclu à l'appel de la procédure
Lors de l'appel de la procédure les paramétres réel remplacent les paramétres formels créant ainsi un mécanisme d'échange d'information entre la procédure ey son point de référance

```
PROCEDURE (x,y)
 DEBUT
  si x>y
   alors
    MAX=x

   si non
    MAX=y
  fin si
 écrire("le maximum entre ",x,"et",y,"est",MAX)
FIN

ALGORITHME       Maximum
DEBUT
 écrire("
deonner deux valeurs distinctes")
 REPETER
  lire(A,B)
 JUSQU'A(A<>B)
 maxi(A,B)
FIN
```

A,B:deux parametres réels
x,y:deux paramétre formels
MAX:variable locale
PASSAGE DES PARAMETRES PAR VALEURS
Un paramétre par valeur est considéré comme un paramétre d'entré
Le sens de transfert des données se fait a une seule direction du paramétre réel en paramétre formel
L'emploi d'un paramétre par valeur implique le transfert d'une valeur plutôt que la substitution d'un paramétre réel
=Lors du transfert on assigne la valeur du paramétre réel au paramétre formel par valeur
exemple:

```
PROCEDURE    UNE(x,y)
 DEBUT
  x=x+5
  Y=y*2
  écrire ("x=",x,"y=",y)
FIN

ALGORITHME      VALEUR
 DEBUT
   lire("A=",A,"B=",B)
```

UNE(A,B)
écrire("A=",A,"B=",B)
FIN
Passage par valeur du programme principal vers la procédure

	PROGRAMME PRINCIPAL
A	5
B	8

PROCEDURE

x• 5 10• • y• 8 16• • A=5 B=8
• 5 10• • y• 8 16• • A=5 B=8
5 10• • y• 8 16• • A=5 B=8
10• • y• 8 16• • A=5 B=8
10• • y• 8 16• • A=5 B=8
10• • y• 8 16• • A=5 B=8
0• • y• 8 16• • A=5 B=8
• • y• 8 16• • A=5 B=8
• y• 8 16• • A=5 B=8
y• 8 16• • A=5 B=8
• 8 16• • A=5 B=8
8 16• • A=5 B=8
16• • A=5 B=8
16• • A=5 B=8
16• • A=5 B=8
6• • A=5 B=8
• • A=5 B=8
• A=5 B=8
A=5 B=8
=5 B=8
5 B=8
B=8
B=8
B=8
B=8
B=8
B=8
B=8
B=8
=8
8

x=10 y=16
=10 y=16
10 y=16
0 y=16
y=16
y=16
y=16
y=16
y=16

y=16
y=16
=16
16
6

A=5 B=8
=5 B=8
5 B=8
B=8
B=8
B=8
B=8
B=8
B=8
B=8
B=8
=8
8

1) Le nombre de paramétre réel doit étre égale au nombre de paramétre formel
) Le nombre de paramétre réel doit étre égale au nombre de paramétre formel
Le nombre de paramétre réel doit étre égale au nombre de paramétre formel
Le nombre de paramétre réel doit étre égale au nombre de paramétre formel
e nombre de paramétre réel doit étre égale au nombre de paramétre formel
nombre de paramétre réel doit étre égale au nombre de paramétre formel
nombre de paramétre réel doit étre égale au nombre de paramétre formel
ombre de paramétre réel doit étre égale au nombre de paramétre formel
mbre de paramétre réel doit étre égale au nombre de paramétre formel
bre de paramétre réel doit étre égale au nombre de paramétre formel
re de paramétre réel doit étre égale au nombre de paramétre formel
e de paramétre réel doit étre égale au nombre de paramétre formel
de paramétre réel doit étre égale au nombre de paramétre formel
de paramétre réel doit étre égale au nombre de paramétre formel
e paramétre réel doit étre égale au nombre de paramétre formel
paramétre réel doit étre égale au nombre de paramétre formel
paramétre réel doit étre égale au nombre de paramétre formel
aramétre réel doit étre égale au nombre de paramétre formel
ramétre réel doit étre égale au nombre de paramétre formel
amétre réel doit étre égale au nombre de paramétre formel
métre réel doit étre égale au nombre de paramétre formel
étre réel doit étre égale au nombre de paramétre formel
tre réel doit étre égale au nombre de paramétre formel
re réel doit étre égale au nombre de paramétre formel
e réel doit étre égale au nombre de paramétre formel
réel doit étre égale au nombre de paramétre formel
réel doit étre égale au nombre de paramétre formel
éel doit étre égale au nombre de paramétre formel
el doit étre égale au nombre de paramétre formel
l doit étre égale au nombre de paramétre formel

doit étre égale au nombre de paramétre formel
doit étre égale au nombre de paramétre formel
oit étre égale au nombre de paramétre formel
it étre égale au nombre de paramétre formel
t étre égale au nombre de paramétre formel
étre égale au nombre de paramétre formel
étre égale au nombre de paramétre formel
tre égale au nombre de paramétre formel
re égale au nombre de paramétre formel
e égale au nombre de paramétre formel
égale au nombre de paramétre formel
égale au nombre de paramétre formel
gale au nombre de paramétre formel
ale au nombre de paramétre formel
le au nombre de paramétre formel
e au nombre de paramétre formel
au nombre de paramétre formel
au nombre de paramétre formel
u nombre de paramétre formel
nombre de paramétre formel
nombre de paramétre formel
ombre de paramétre formel
mbre de paramétre formel
bre de paramétre formel
re de paramétre formel
e de paramétre formel
de paramétre formel
de paramétre formel
e paramétre formel
paramétre formel
paramétre formel
aramétre formel
ramétre formel
amétre formel
métre formel
étre formel
tre formel
re formel
e formel
formel
formel
ormel
rmel
mel
el
l

2) Les paramétres réels et formels doivent être de mème type
) Les paramétres réels et formels doivent être de mème type
Les paramétres réels et formels doivent être de mème type

Les paramétres réels et formels doivent être de mème type
es paramétres réels et formels doivent être de mème type
s paramétres réels et formels doivent être de mème type
paramétres réels et formels doivent être de mème type
paramétres réels et formels doivent être de mème type
aramétres réels et formels doivent être de mème type
ramétres réels et formels doivent être de mème type
amétres réels et formels doivent être de mème type
métres réels et formels doivent être de mème type
étres réels et formels doivent être de mème type
tres réels et formels doivent être de mème type
res réels et formels doivent être de mème type
es réels et formels doivent être de mème type
s réels et formels doivent être de mème type
réels et formels doivent être de mème type
réels et formels doivent être de mème type
éels et formels doivent être de mème type
els et formels doivent être de mème type
ls et formels doivent être de mème type
s et formels doivent être de mème type
et formels doivent être de mème type
et formels doivent être de mème type
t formels doivent être de mème type
formels doivent être de mème type
formels doivent être de mème type
ormels doivent être de mème type
rmels doivent être de mème type
mels doivent être de mème type
els doivent être de mème type
ls doivent être de mème type
s doivent être de mème type
doivent être de mème type
doivent être de mème type
oivent être de mème type
ivent être de mème type
vent être de mème type
ent être de mème type
nt être de mème type
t être de mème type
être de mème type
être de mème type
tre de mème type
re de mème type
e de mème type
de mème type
de mème type
e mème type
mème type
mème type
ème type

me type
e type
 type
type
ype
pe
e

3) L'ordre dans le transfert intervient
) L'ordre dans le transfert intervient
 L'ordre dans le transfert intervient
L'ordre dans le transfert intervient
'ordre dans le transfert intervient
ordre dans le transfert intervient
rdre dans le transfert intervient
dre dans le transfert intervient
re dans le transfert intervient
e dans le transfert intervient
 dans le transfert intervient
dans le transfert intervient
ans le transfert intervient
ns le transfert intervient
s le transfert intervient
 le transfert intervient
le transfert intervient
e transfert intervient
 transfert intervient
transfert intervient
ransfert intervient
ansfert intervient
nsfert intervient
sfert intervient
fert intervient
ert intervient
rt intervient
t intervient
 intervient
intervient
ntervient
tervient
ervient
rvient
vient
ient
ent
nt
t

Passage des paramétres par variable ou réfèrence
assage des paramétres par variable ou réfèrence

ssage des paramétres par variable ou réfèrence
sage des paramétres par variable ou réfèrence
age des paramétres par variable ou réfèrence
ge des paramétres par variable ou réfèrence
e des paramétres par variable ou réfèrence
des paramétres par variable ou réfèrence
des paramétres par variable ou réfèrence
es paramétres par variable ou réfèrence
s paramétres par variable ou réfèrence
paramétres par variable ou réfèrence
paramétres par variable ou réfèrence
aramétres par variable ou réfèrence
ramétres par variable ou réfèrence
amétres par variable ou réfèrence
métres par variable ou réfèrence
étres par variable ou réfèrence
tres par variable ou réfèrence
res par variable ou réfèrence
es par variable ou réfèrence
s par variable ou réfèrence
par variable ou réfèrence
par variable ou réfèrence
ar variable ou réfèrence
r variable ou réfèrence
variable ou réfèrence
variable ou réfèrence
ariable ou réfèrence
riable ou réfèrence
iable ou réfèrence
able ou réfèrence
ble ou réfèrence
le ou réfèrence
e ou réfèrence
ou réfèrence
ou réfèrence
u réfèrence
réfèrence
réfèrence
éfèrence
fèrence
èrence
rence
ence
nce
ce
e

1)Un paramétre variable est passé par réfèrence et non par valeur
)Un paramétre variable est passé par réfèrence et non par valeur

Un paramétre variable est passé par réfèrence et non par valeur
n paramétre variable est passé par réfèrence et non par valeur
paramétre variable est passé par réfèrence et non par valeur
paramétre variable est passé par réfèrence et non par valeur
aramétre variable est passé par réfèrence et non par valeur
ramétre variable est passé par réfèrence et non par valeur
amétre variable est passé par réfèrence et non par valeur
métre variable est passé par réfèrence et non par valeur
étre variable est passé par réfèrence et non par valeur
tre variable est passé par réfèrence et non par valeur
re variable est passé par réfèrence et non par valeur
e variable est passé par réfèrence et non par valeur
variable est passé par réfèrence et non par valeur
variable est passé par réfèrence et non par valeur
ariable est passé par réfèrence et non par valeur
riable est passé par réfèrence et non par valeur
iable est passé par réfèrence et non par valeur
able est passé par réfèrence et non par valeur
ble est passé par réfèrence et non par valeur
le est passé par réfèrence et non par valeur
e est passé par réfèrence et non par valeur
est passé par réfèrence et non par valeur
est passé par réfèrence et non par valeur
st passé par réfèrence et non par valeur
t passé par réfèrence et non par valeur
passé par réfèrence et non par valeur
passé par réfèrence et non par valeur
assé par réfèrence et non par valeur
ssé par réfèrence et non par valeur
sé par réfèrence et non par valeur
é par réfèrence et non par valeur
par réfèrence et non par valeur
par réfèrence et non par valeur
ar réfèrence et non par valeur
r réfèrence et non par valeur
réfèrence et non par valeur
réfèrence et non par valeur
éfèrence et non par valeur
fèrence et non par valeur
èrence et non par valeur
rence et non par valeur
ence et non par valeur
nce et non par valeur
ce et non par valeur
e et non par valeur
et non par valeur
et non par valeur
t non par valeur
non par valeur
non par valeur

on par valeur
n par valeur
 par valeur
par valeur
ar valeur
r valeur
 valeur
valeur
aleur
leur
eur
ur
r

2) Pour distinguer les paramétres variables des paramétres par valeur on procéde le nom de la
) Pour distinguer les paramétres variables des paramétres par valeur on procéde le nom de la
 Pour distinguer les paramétres variables des paramétres par valeur on procéde le nom de la
Pour distinguer les paramétres variables des paramétres par valeur on procéde le nom de la
our distinguer les paramétres variables des paramétres par valeur on procéde le nom de la
ur distinguer les paramétres variables des paramétres par valeur on procéde le nom de la
r distinguer les paramétres variables des paramétres par valeur on procéde le nom de la
 distinguer les paramétres variables des paramétres par valeur on procéde le nom de la
distinguer les paramétres variables des paramétres par valeur on procéde le nom de la variable
istinguer les paramétres variables des paramétres par valeur on procéde le nom de la variable
stinguer les paramétres variables des paramétres par valeur on procéde le nom de la variable
tinguer les paramétres variables des paramétres par valeur on procéde le nom de la variable
inguer les paramétres variables des paramétres par valeur on procéde le nom de la variable par
nguer les paramétres variables des paramétres par valeur on procéde le nom de la variable par
guer les paramétres variables des paramétres par valeur on procéde le nom de la variable par
uer les paramétres variables des paramétres par valeur on procéde le nom de la variable par le
er les paramétres variables des paramétres par valeur on procéde le nom de la variable par le
r les paramétres variables des paramétres par valeur on procéde le nom de la variable par le
 les paramétres variables des paramétres par valeur on procéde le nom de la variable par le
les paramétres variables des paramétres par valeur on procéde le nom de la variable par le mot
es paramétres variables des paramétres par valeur on procéde le nom de la variable par le mot
s paramétres variables des paramétres par valeur on procéde le nom de la variable par le mot
 paramétres variables des paramétres par valeur on procéde le nom de la variable par le mot
paramétres variables des paramétres par valeur on procéde le nom de la variable par le mot clé
aramétres variables des paramétres par valeur on procéde le nom de la variable par le mot clé
ramétres variables des paramétres par valeur on procéde le nom de la variable par le mot clé
amétres variables des paramétres par valeur on procéde le nom de la variable par le mot clé
métres variables des paramétres par valeur on procéde le nom de la variable par le mot clé var
étres variables des paramétres par valeur on procéde le nom de la variable par le mot clé var
tres variables des paramétres par valeur on procéde le nom de la variable par le mot clé var
res variables des paramétres par valeur on procéde le nom de la variable par le mot clé var (ou
es variables des paramétres par valeur on procéde le nom de la variable par le mot clé var (ou
s variables des paramétres par valeur on procéde le nom de la variable par le mot clé var (ou
 variables des paramétres par valeur on procéde le nom de la variable par le mot clé var (ou
variables des paramétres par valeur on procéde le nom de la variable par le mot clé var (ou
ariables des paramétres par valeur on procéde le nom de la variable par le mot clé var (ou ref)

riables des paramétres par valeur on procéde le nom de la variable par le mot clé var (ou ref)
iables des paramétres par valeur on procéde le nom de la variable par le mot clé var (ou ref)
ables des paramétres par valeur on procéde le nom de la variable par le mot clé var (ou ref)
bles des paramétres par valeur on procéde le nom de la variable par le mot clé var (ou ref)
les des paramétres par valeur on procéde le nom de la variable par le mot clé var (ou ref)
es des paramétres par valeur on procéde le nom de la variable par le mot clé var (ou ref)
s des paramétres par valeur on procéde le nom de la variable par le mot clé var (ou ref)
des paramétres par valeur on procéde le nom de la variable par le mot clé var (ou ref)
des paramétres par valeur on procéde le nom de la variable par le mot clé var (ou ref)
es paramétres par valeur on procéde le nom de la variable par le mot clé var (ou ref)
s paramétres par valeur on procéde le nom de la variable par le mot clé var (ou ref)
paramétres par valeur on procéde le nom de la variable par le mot clé var (ou ref)
paramétres par valeur on procéde le nom de la variable par le mot clé var (ou ref)
aramétres par valeur on procéde le nom de la variable par le mot clé var (ou ref)
ramétres par valeur on procéde le nom de la variable par le mot clé var (ou ref)
amétres par valeur on procéde le nom de la variable par le mot clé var (ou ref)
métres par valeur on procéde le nom de la variable par le mot clé var (ou ref)
étres par valeur on procéde le nom de la variable par le mot clé var (ou ref)
tres par valeur on procéde le nom de la variable par le mot clé var (ou ref)
res par valeur on procéde le nom de la variable par le mot clé var (ou ref)
es par valeur on procéde le nom de la variable par le mot clé var (ou ref)
s par valeur on procéde le nom de la variable par le mot clé var (ou ref)
par valeur on procéde le nom de la variable par le mot clé var (ou ref)
par valeur on procéde le nom de la variable par le mot clé var (ou ref)
ar valeur on procéde le nom de la variable par le mot clé var (ou ref)
r valeur on procéde le nom de la variable par le mot clé var (ou ref)
valeur on procéde le nom de la variable par le mot clé var (ou ref)
valeur on procéde le nom de la variable par le mot clé var (ou ref)
aleur on procéde le nom de la variable par le mot clé var (ou ref)
leur on procéde le nom de la variable par le mot clé var (ou ref)
eur on procéde le nom de la variable par le mot clé var (ou ref)
ur on procéde le nom de la variable par le mot clé var (ou ref)
r on procéde le nom de la variable par le mot clé var (ou ref)
on procéde le nom de la variable par le mot clé var (ou ref)
on procéde le nom de la variable par le mot clé var (ou ref)
n procéde le nom de la variable par le mot clé var (ou ref)
procéde le nom de la variable par le mot clé var (ou ref)
procéde le nom de la variable par le mot clé var (ou ref)
rocéde le nom de la variable par le mot clé var (ou ref)
océde le nom de la variable par le mot clé var (ou ref)
céde le nom de la variable par le mot clé var (ou ref)
éde le nom de la variable par le mot clé var (ou ref)
de le nom de la variable par le mot clé var (ou ref)
e le nom de la variable par le mot clé var (ou ref)
le nom de la variable par le mot clé var (ou ref)
le nom de la variable par le mot clé var (ou ref)
e nom de la variable par le mot clé var (ou ref)
nom de la variable par le mot clé var (ou ref)
nom de la variable par le mot clé var (ou ref)
om de la variable par le mot clé var (ou ref)

m de la variable par le mot clé var (ou ref)
de la variable par le mot clé var (ou ref)
de la variable par le mot clé var (ou ref)
e la variable par le mot clé var (ou ref)
la variable par le mot clé var (ou ref)
la variable par le mot clé var (ou ref)
a variable par le mot clé var (ou ref)
variable par le mot clé var (ou ref)
variable par le mot clé var (ou ref)
ariable par le mot clé var (ou ref)
riable par le mot clé var (ou ref)
iable par le mot clé var (ou ref)
able par le mot clé var (ou ref)
ble par le mot clé var (ou ref)
le par le mot clé var (ou ref)
e par le mot clé var (ou ref)
par le mot clé var (ou ref)
par le mot clé var (ou ref)
ar le mot clé var (ou ref)
r le mot clé var (ou ref)
le mot clé var (ou ref)
le mot clé var (ou ref)
e mot clé var (ou ref)
mot clé var (ou ref)
mot clé var (ou ref)
ot clé var (ou ref)
t clé var (ou ref)
clé var (ou ref)
clé var (ou ref)
lé var (ou ref)
é var (ou ref)
var (ou ref)
var (ou ref)
ar (ou ref)
r (ou ref)
(ou ref)
(ou ref)
ou ref)
u ref)
ref)
ref)
ef)
f)
)

3)Unparamétre variable ne peut être qu'une variable et non une cte ou expression
)Unparamétre variable ne peut être qu'une variable et non une cte ou expression
Unparamétre variable ne peut être qu'une variable et non une cte ou expression
nparamétre variable ne peut être qu'une variable et non une cte ou expression
paramétre variable ne peut être qu'une variable et non une cte ou expression

aramétre variable ne peut être qu'une variable et non une cte ou expression
ramétre variable ne peut être qu'une variable et non une cte ou expression
amétre variable ne peut être qu'une variable et non une cte ou expression
métre variable ne peut être qu'une variable et non une cte ou expression
étre variable ne peut être qu'une variable et non une cte ou expression
tre variable ne peut être qu'une variable et non une cte ou expression
re variable ne peut être qu'une variable et non une cte ou expression
e variable ne peut être qu'une variable et non une cte ou expression
variable ne peut être qu'une variable et non une cte ou expression
variable ne peut être qu'une variable et non une cte ou expression
ariable ne peut être qu'une variable et non une cte ou expression
riable ne peut être qu'une variable et non une cte ou expression
iable ne peut être qu'une variable et non une cte ou expression
able ne peut être qu'une variable et non une cte ou expression
ble ne peut être qu'une variable et non une cte ou expression
le ne peut être qu'une variable et non une cte ou expression
e ne peut être qu'une variable et non une cte ou expression
ne peut être qu'une variable et non une cte ou expression
ne peut être qu'une variable et non une cte ou expression
e peut être qu'une variable et non une cte ou expression
peut être qu'une variable et non une cte ou expression
peut être qu'une variable et non une cte ou expression
eut être qu'une variable et non une cte ou expression
ut être qu'une variable et non une cte ou expression
t être qu'une variable et non une cte ou expression
être qu'une variable et non une cte ou expression
être qu'une variable et non une cte ou expression
tre qu'une variable et non une cte ou expression
re qu'une variable et non une cte ou expression
e qu'une variable et non une cte ou expression
qu'une variable et non une cte ou expression
qu'une variable et non une cte ou expression
u'une variable et non une cte ou expression
'une variable et non une cte ou expression
une variable et non une cte ou expression
ne variable et non une cte ou expression
e variable et non une cte ou expression
variable et non une cte ou expression
variable et non une cte ou expression
ariable et non une cte ou expression
riable et non une cte ou expression
iable et non une cte ou expression
able et non une cte ou expression
ble et non une cte ou expression
le et non une cte ou expression
e et non une cte ou expression
et non une cte ou expression
et non une cte ou expression
t non une cte ou expression
non une cte ou expression

non une cte ou expression
on une cte ou expression
n une cte ou expression
une cte ou expression
une cte ou expression
ne cte ou expression
e cte ou expression
cte ou expression
cte ou expression
te ou expression
e ou expression
ou expression
ou expression
u expression
expression
expression
xpression
pression
ression
ession
ssion
sion
ion
on
n

4) changement dans la valeur d’un paramétre formel à l’intérieur d’une procedure
) changement dans la valeur d’un paramétre formel à l’intérieur d’une procedure changera
changement dans la valeur d’un paramétre formel à l’intérieur d’une procedure changera
changement dans la valeur d’un paramétre formel à l’intérieur d’une procedure changera
changement dans la valeur d’un paramétre formel à l’intérieur d’une procedure changera
changement dans la valeur d’un paramétre formel à l’intérieur d’une procedure changera
changement dans la valeur d’un paramétre formel à l’intérieur d’une procedure changera
changement dans la valeur d’un paramétre formel à l’intérieur d’une procedure changera
changement dans la valeur d’un paramétre formel à l’intérieur d’une procedure changera
changement dans la valeur d’un paramétre formel à l’intérieur d’une procedure changera
changement dans la valeur d’un paramétre formel à l’intérieur d’une procedure changera
hangement dans la valeur d’un paramétre formel à l’intérieur d’une procedure changera
angement dans la valeur d’un paramétre formel à l’intérieur d’une procedure changera
ngement dans la valeur d’un paramétre formel à l’intérieur d’une procedure changera
gement dans la valeur d’un paramétre formel à l’intérieur d’une procedure changera
ement dans la valeur d’un paramétre formel à l’intérieur d’une procedure changera également
ment dans la valeur d’un paramétre formel à l’intérieur d’une procedure changera également
ent dans la valeur d’un paramétre formel à l’intérieur d’une procedure changera également la
nt dans la valeur d’un paramétre formel à l’intérieur d’une procedure changera également la
t dans la valeur d’un paramétre formel à l’intérieur d’une procedure changera également la
dans la valeur d’un paramétre formel à l’intérieur d’une procedure changera également la
dans la valeur d’un paramétre formel à l’intérieur d’une procedure changera également la
ans la valeur d’un paramétre formel à l’intérieur d’une procedure changera également la

ns la valeur d’un paramétre formel à l’intérieur d’une procedure changera également la valeur
s la valeur d’un paramétre formel à l’intérieur d’une procedure changera également la valeur
la valeur d’un paramétre formel à l’intérieur d’une procedure changera également la valeur
la valeur d’un paramétre formel à l’intérieur d’une procedure changera également la valeur du
a valeur d’un paramétre formel à l’intérieur d’une procedure changera également la valeur du
valeur d’un paramétre formel à l’intérieur d’une procedure changera également la valeur du
valeur d’un paramétre formel à l’intérieur d’une procedure changera également la valeur du
aleur d’un paramétre formel à l’intérieur d’une procedure changera également la valeur du
leur d’un paramétre formel à l’intérieur d’une procedure changera également la valeur du
eur d’un paramétre formel à l’intérieur d’une procedure changera également la valeur du
ur d’un paramétre formel à l’intérieur d’une procedure changera également la valeur du
r d’un paramétre formel à l’intérieur d’une procedure changera également la valeur du
d’un paramétre formel à l’intérieur d’une procedure changera également la valeur du
d’un paramétre formel à l’intérieur d’une procedure changera également la valeur du
’un paramétre formel à l’intérieur d’une procedure changera également la valeur du paramétre
un paramétre formel à l’intérieur d’une procedure changera également la valeur du paramétre
n paramétre formel à l’intérieur d’une procedure changera également la valeur du paramétre
paramétre formel à l’intérieur d’une procedure changera également la valeur du paramétre
paramétre formel à l’intérieur d’une procedure changera également la valeur du paramétre
aramétre formel à l’intérieur d’une procedure changera également la valeur du paramétre réel
ramétre formel à l’intérieur d’une procedure changera également la valeur du paramétre réel
amétre formel à l’intérieur d’une procedure changera également la valeur du paramétre réel
métre formel à l’intérieur d’une procedure changera également la valeur du paramétre réel
étre formel à l’intérieur d’une procedure changera également la valeur du paramétre réel
tre formel à l’intérieur d’une procedure changera également la valeur du paramétre réel
re formel à l’intérieur d’une procedure changera également la valeur du paramétre réel
e formel à l’intérieur d’une procedure changera également la valeur du paramétre réel
formel à l’intérieur d’une procedure changera également la valeur du paramétre réel
formel à l’intérieur d’une procedure changera également la valeur du paramétre réel
ormel à l’intérieur d’une procedure changera également la valeur du paramétre réel
rmel à l’intérieur d’une procedure changera également la valeur du paramétre réel
mel à l’intérieur d’une procedure changera également la valeur du paramétre réel
el à l’intérieur d’une procedure changera également la valeur du paramétre réel correspondant
l à l’intérieur d’une procedure changera également la valeur du paramétre réel correspondant à
à l’intérieur d’une procedure changera également la valeur du paramétre réel correspondant à
à l’intérieur d’une procedure changera également la valeur du paramétre réel correspondant à
l’intérieur d’une procedure changera également la valeur du paramétre réel correspondant à
l’intérieur d’une procedure changera également la valeur du paramétre réel correspondant à
’intérieur d’une procedure changera également la valeur du paramétre réel correspondant à
intérieur d’une procedure changera également la valeur du paramétre réel correspondant à
ntérieur d’une procedure changera également la valeur du paramétre réel correspondant à
térieur d’une procedure changera également la valeur du paramétre réel correspondant à
érieur d’une procedure changera également la valeur du paramétre réel correspondant à
rieur d’une procedure changera également la valeur du paramétre réel correspondant à
ieur d’une procedure changera également la valeur du paramétre réel correspondant à
eur d’une procedure changera également la valeur du paramétre réel correspondant à
ur d’une procedure changera également la valeur du paramétre réel correspondant à
r d’une procedure changera également la valeur du paramétre réel correspondant à l’extérieur
d’une procedure changera également la valeur du paramétre réel correspondant à l’extérieur
d’une procedure changera également la valeur du paramétre réel correspondant à l’extérieur

’une procedure changera également la valeur du paramétre réel correspondant à l’extérieur de
une procedure changera également la valeur du paramétre réel correspondant à l’extérieur de
ne procedure changera également la valeur du paramétre réel correspondant à l’extérieur de
e procedure changera également la valeur du paramétre réel correspondant à l’extérieur de
procedure changera également la valeur du paramétre réel correspondant à l’extérieur de
procedure changera également la valeur du paramétre réel correspondant à l’extérieur de
rocedure changera également la valeur du paramétre réel correspondant à l’extérieur de
ocedure changera également la valeur du paramétre réel correspondant à l’extérieur de
cedure changera également la valeur du paramétre réel correspondant à l’extérieur de
edure changera également la valeur du paramétre réel correspondant à l’extérieur de
dure changera également la valeur du paramétre réel correspondant à l’extérieur de
ure changera également la valeur du paramétre réel correspondant à l’extérieur de
re changera également la valeur du paramétre réel correspondant à l’extérieur de
e changera également la valeur du paramétre réel correspondant à l’extérieur de
changera également la valeur du paramétre réel correspondant à l’extérieur de
changera également la valeur du paramétre réel correspondant à l’extérieur de
hangera également la valeur du paramétre réel correspondant à l’extérieur de
angera également la valeur du paramétre réel correspondant à l’extérieur de
ngera également la valeur du paramétre réel correspondant à l’extérieur de
gera également la valeur du paramétre réel correspondant à l’extérieur de
era également la valeur du paramétre réel correspondant à l’extérieur de
ra également la valeur du paramétre réel correspondant à l’extérieur de
a également la valeur du paramétre réel correspondant à l’extérieur de
également la valeur du paramétre réel correspondant à l’extérieur de
également la valeur du paramétre réel correspondant à l’extérieur de
galement la valeur du paramétre réel correspondant à l’extérieur de
alement la valeur du paramétre réel correspondant à l’extérieur de
lement la valeur du paramétre réel correspondant à l’extérieur de
ement la valeur du paramétre réel correspondant à l’extérieur de
ment la valeur du paramétre réel correspondant à l’extérieur de
ent la valeur du paramétre réel correspondant à l’extérieur de
nt la valeur du paramétre réel correspondant à l’extérieur de
t la valeur du paramétre réel correspondant à l’extérieur de
la valeur du paramétre réel correspondant à l’extérieur de
la valeur du paramétre réel correspondant à l’extérieur de
a valeur du paramétre réel correspondant à l’extérieur de
valeur du paramétre réel correspondant à l’extérieur de
valeur du paramétre réel correspondant à l’extérieur de
aleur du paramétre réel correspondant à l’extérieur de
leur du paramétre réel correspondant à l’extérieur de
eur du paramétre réel correspondant à l’extérieur de
ur du paramétre réel correspondant à l’extérieur de
r du paramétre réel correspondant à l’extérieur de
du paramétre réel correspondant à l’extérieur de
du paramétre réel correspondant à l’extérieur de
u paramétre réel correspondant à l’extérieur de
paramétre réel correspondant à l’extérieur de
paramétre réel correspondant à l’extérieur de
aramétre réel correspondant à l’extérieur de
ramétre réel correspondant à l’extérieur de

amétre réel correspondant à l'extérieur de
métre réel correspondant à l'extérieur de
étre réel correspondant à l'extérieur de
tre réel correspondant à l'extérieur de
re réel correspondant à l'extérieur de
e réel correspondant à l'extérieur de
réel correspondant à l'extérieur de
réel correspondant à l'extérieur de
éel correspondant à l'extérieur de
el correspondant à l'extérieur de
l correspondant à l'extérieur de
correspondant à l'extérieur de
correspondant à l'extérieur de
orrespondant à l'extérieur de
rrespondant à l'extérieur de
respondant à l'extérieur de
espondant à l'extérieur de
spondant à l'extérieur de
pondant à l'extérieur de
ondant à l'extérieur de
ndant à l'extérieur de
dant à l'extérieur de
ant à l'extérieur de
nt à l'extérieur de
t à l'extérieur de
à l'extérieur de
à l'extérieur de
l'extérieur de
l'extérieur de
'extérieur de
extérieur de
xtérieur de
térieur de
érieur de
rieur de
ieur de
eur de
ur de
r de
de
de
e

Algorithme échange
Début
Introduire la valeur de A et introduire la valeur de B
Introduire la valeur de A ⇨lire (A) →
Introduire la valeur de B ⇨ lire (B)
Affecté la valeur de A dans l'auxiliaire ⇨ auxiliaire ←A
Affecté la valeur de B dans A ⇨ A←B
Affecté la valeur de l'auxiliaire dans B ⇨ B ←auxiliaire
Fin

QUELQUE TERMINOLOGIE UTILISE DE LA NOTION ALGORITHME

Processeur : est toute entité capable de comprendre un énoncé et d'exécuter le travail indiqué par cet énoncé .

Environnement : l'ensemble des objectifs nécessaire l'exécution d'un travail .

Action primitive : une action est primitive si l'énonce de cette action à lui seul suffisant pour que le processeur puisse l'exécuter sans information supplémentaire ; une action non primitive doivent décomposer en action primitive. Pour composer une action non primitive, on peut utiliser plusieurs approche tel que l'analyse descendante.

Etant donné un travail T décrit pour un énoncé non primitive ; l'analyse descendante constitue à trouver une décomposition de T à seule d'énonce .

LES CONSTANTES ET LES VARIABLES

Nous considérons l'action suivant

Ajouter 1 à la valeur de x et affecter le résultat comme novelle valeur de x , cette exemple fait intervenir deux objets 1 et x .

L'objet x soit sa valeur varier ⇨ on dit que x est un **variable,** et nous dirons que 1 est un **constant .**

Définition : 1-une variable est un objet dont la valeur n'est pas invariable (c'est peut modifier au cours d'une exécution).

Toute variable est défini par :

-un nom qui sert à designer et qui commence par une par une par une lettre

-un type qui décrit l'utilisation possible de la valeur .

Définir variable c'est en fait créer un objet pour le processeur .

2-une constante est un objet de valeur invariable . Exemple : x←1

Action d'affectation : constante ←variable Exemple : x←y

Les types de donnés élémentaires :

1-*Le type numérique :* c'est l'ensemble des valeurs numériques . Une valeur numérique sera écrite sous sa forme habituelle avec ou sans signe(une valeur sans signe est considéré comme positive).

2-*Le type caractère :* c'est l'ensemble des chaines que l'on peut former à partir des éléments de l'ensemble des caractères(lettres , chiffres et spéciaux) .

Ce type est connu sous le nom type alpha-numérique .

Pour éviter de confondre une chaine de caractère avec le nom d'une variable , on le représentera entre deux apostrophes qui délimitent le début et la fin de la chaine et ne seront pas considères comme faisant partie de la cheine .

LES PROCEDURES ET LES FONCTION

DEFINITION

Une procedure est un sous programme realisent un traitement sur une partie du donnés d'un programme principal

exemple : ecrire un programme(alg) qui effectue les traitement suivant

-)saisie des notes de chaque éleve = procédure

-)calcul de la moyenne = procedure

-)cllassement=procédure est qu'elle évite l'ecriture de la même chose plusieur fois

POUR QUOI UNE PROCEDURE :

Les procédures permettent de decomposer un algorithme en entité plus simple et donc de simplifier la lecture et le suivi d'un algorithme

un procédure peut etre appelée à :
-)partir de l'algorithme principal
ou
-)partir d'une entre procédure
-)une procédure peut retourner ou non une valeur comme elle peut procéder ses propres variables ou peut utiliser les variables de l'algorithme appelant
exemple :
calcul de la combinaison
on remarque qu'il ya une traitement de calcul du factoriel qui se répetes 3 fois
avec l'utilisation de la procédure,on écrit une procédure qui calcule un factoriel et on l'apple 3 fois
-)une fois pour(n)
-)une fois pour(p)
-)une fois pour(p - n)
l'appel de la procédure se fait tout simplement en faisant réference à son nom suivi ou non de ses paramétres séparés par des virgules et entourés par des ( )
Quand on appelle une procédure le controle se trouve automatiquement transféré au début de la procédure
Les instructions d'action à l'interieur de la procédure s'exécutent en prenant en considérons toutes les variables propres à la procédure.
Quand toutes les actions de la procédure ont été exécutées le contrôle retourne à l'action qui suit immédiatement l'action d'appel de la procédure.
Une procédure est finie par son nom avec ou sans paramétres
exemple :
procédure sans paramétre
procédure méssage
DEBUT
écrire(" bongour ")
FIN
ALGORITHME PRINCIPALE
DEBUT
message
FIN
procédure avec paramétre
procédure MESSAGE(M)
DEBUT
écrire(" bongour ",)
FIN
ALGORITHME PRINCIPAL
DEBUT
MESSAGE(" monsieur ")
MESSAGE(" madame ")
FINT
OU
ALGORITHME PRINCIPAL3
DEBUT
lire(personne)
MESSAGE(personne)
FIN
L'EXERCICE 7 DE LA SERIE N 2

Une société fabrique des objets en plastique qu’elle peut emballer dans les caisses de différentes capacité s

caisse G	213 unités
caisse M	36 unités
caisse P1	6 unités
caisse p2	1 nités

```
ALGORITHME    SANS-PROCEDURE
DEBUT
  écrire(" donner la valeur de la quantité commandée?")
  REPETER
    LIRE (Q)
  JUSQU'A(Q<10000)
  G=E(Q/213)
  M=E(Q-G*213)/316)
  p1=E(Q-G*213)-M*36)/G)
  P2=E(Q-G*213-M*36-P1*6)
   si(G<>0)
    alors
     écrire("le nombre de caisse type G est:"G)
   fin si
   si(M<>0)
    alors
     écrire("le nombre de caisse type M est:"M)
   fin si
   si(P1<>0)
    alors
     écrire("le nombre de caisse type P1 est:"P1)
   fin si
   si(P2<>0)
    alors
     écrire("le nombre de caisse type P2 est:"P2)
   fin si
FIN

ALGORITHME    AVEC-PROCEDURE
 DEBUT
  écrire("donner le valeur de la quantité commandée?")
  REPETER
   LIRE(Q)
  JUSQU'A(Q<10000)
  G=E(Q/213)
  M=E((Q-G*213)/36)
  P1=E((Q-G*213-M*36)/G)
  P2=E(Q-G*213-M*36-P1*6)
  affichage("G,G)
  affichage("M,M)
  affichage("P1,P1)
  affichage("P2,P2)
FIN

PROCEDURE      AFFICHAGE(G,A)
 debut
  si(A<>0)
   alors
    écrire("le nombre de caisse type,"G",est,"A)
```

```
  fin si
FIN
```